

JavaScript and Ajax

http://www.w3schools.com/js/js_examples.asp



open.michigan

Unless otherwise noted, the content of this course material is licensed under a Creative Commons Attribution 3.0 License. <http://creativecommons.org/licenses/by/3.0/>.

Copyright 2009, Charles Severance, Jim Eng





HTML JavaScript
AJAX CSS
Cookies

HTTP Request
Response GET
POST

Python Data Store
Templates memcache
WebApp MVC

JavaScript

- In addition to HTML and CSS...
- Browsers have a **powerful programming** language called JavaScript that runs **in the browser**
- Actually not much like Java - more like C
- Very powerful and flexible - we keep “discovering” new power

<http://en.wikipedia.org/wiki/JavaScript>

Language Syntax

- Whitespace does not matter - spaces and new lines
- Begin and end of blocks are curly braces
- Statements must end in semicolons

```
function message()  
{  
    alert("This alert box was called with the onload event");  
}
```

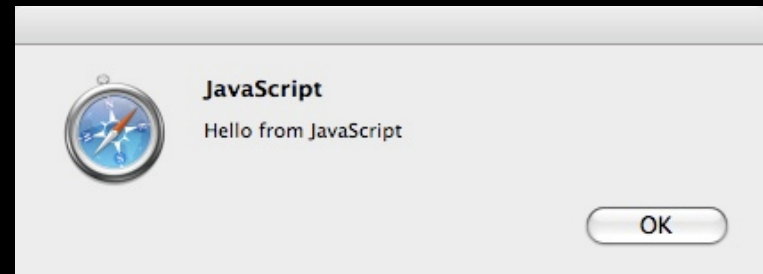
Language Syntax

- Javascript supports **comment** characters that start and end a **comment** and can be multiline as well as a **comment** character that **comments** to the end of the current line

```
function message()  
{  
    /* I am a  
       multi-line comment */  
    alert("This alert box was called with the onload event");  
    // I am a comment to the end of one line  
}
```

The Most Useful JavaScript

- The alert box is often the way we debug simple JavaScript
- When it runs - alert makes a pop up box



```
alert("Hello from JavaScript");
```

Including JavaScript in a Page

- Include an File from a URL
- Inline Code
- As an Attribute on a Tag

Including a File from a URL

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>App Engine - HTML</title>
    <link href="/static/glike.css" rel="stylesheet" type="text/css" />
    <script type="text/javascript" src="/static/js/jquery-1.2.6.min.js"></script>
  </head>
  <body>
    <div id="header">
```

Inline Javascript

```
<html>
<head>
</head>
<body>
<h1>Here is my Document</h1>
<script type="text/javascript">
  alert("Hello from JavaScript");
</script>
<h1>Here is my second Header</h1>
</body>
</html>
```

Inline JavaScript
runs as the page
loads.

inline.htm

Validating Inline Javascript

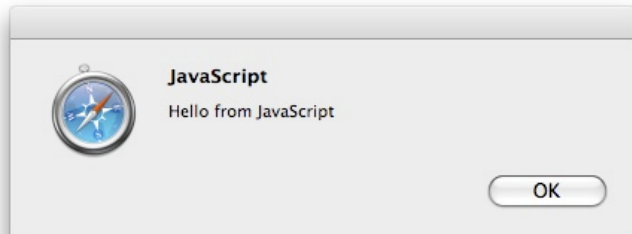
```
<html>
<head>
</head>
<body>
<h1>Here is my Document</h1>
<script type="text/javascript">
  /*  */
  alert("Hello from JavaScript");
  /*  */
</script>
<h1>Here is my second Header</h1>
</body>
</html>
```

We add Javascript
comments and a
CDATA tag to pass
XHTML and
HTML4 Strict
Validation

inline.htm



Here is my Document



Here is my Document

Here is my second Header

```
<html>
<head>
</head>
<body>
<h1>Here is my Document</h1>
<script type="text/javascript">
    alert("Hello from JavaScript");
</script>
<h1>Here is my second Header</h1>
</body>
</html>
```

<http://www.dr-chuck.com/javascript/one.htm>

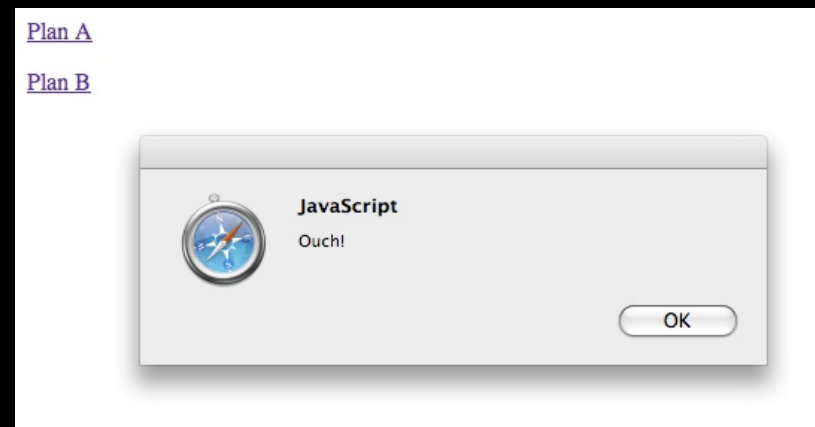
Event Handling

- Certain tags have **attributes** which contain JavaScript which run when things (**events**) “happen” to the tag
- http://www.w3schools.com/jsref/jsref_events.asp
- onchange, onclick, onmousedown, onmouseup ..

```
<a href="http://www.umich.edu" onclick="alert('Ouch!')">Plan A</a>
```

```
<body>
<p>
<a href="http://www.umich.edu" onclick="alert('Ouch!')">
Plan A</a>
</p><p>
<a href="http://www.umich.edu" onclick="alert('Yow!');return false;">
Plan B</a>
</p>
</body>
```

Returning **false** in these events means “don’t do what you were going to do”. In this case, **Plan B** will not follow the link after the popup.



<http://www.dr-chuck.com/javascript/two.htm>

<http://www.dr-chuck.com/javascript/three.htm>

```
<html>
<head>
<script type="text/javascript">
function message()
{
    alert("This alert box was called with the onload event");
}
</script>
</head>
<body onload="message()">
<h1>Hello World</h1>
</body>
</html>
```

Another
Event

http://www.w3schools.com/js/tryit.asp?filename=tryjs_headsection

```
<form method="post" action="/apply">
  <p>
    <label for="name">Name:</label>
    <input type="text" name="name" id="name"/>
  </p>
  <p>
    <label for="account"> Account:</label>
    <input type="text" name="account" id="account"/>
  </p>
  <p>
    <label for="password"> Password:</label>
    <input type="password" name="password" id="password"/>
  </p>
  <input type="submit" name="Login"/>
  <input type="button" onclick="window.location='/'; return false;"
value="Cancel" />
</form>
```



```
<input type="button" onclick="window.location='/'; return false;"  
value="Cancel" />
```

When this button is clicked, navigate this window's location to the “/” URL, and do not submit this form.

Document Object Model

http://en.wikipedia.org/wiki/Document_Object_Model

Document Object Model

- JavaScript can manipulate the current HTML document
- The “Document Object Model” tells us the syntax to use to access various “bits” of the current screen to read and/or manipulate
- You can even find pieces of the model by id attribute and change them

http://en.wikipedia.org/wiki/Document_Object_Model

```
<input type="button" onclick="window.location='/'; return false;"  
      value="Cancel" />
```

When this button is clicked, go into the document model at `window.location` and change it to be `“/”`, and do not submit this form.

<http://www.dr-chuck.com/javascript/four.htm>

```
<p>
<a href="#"
  onclick="document.getElementById('stuff').innerHTML = 'BACK';">BACK</a>
<a href="#"
  onclick="document.getElementById('stuff').innerHTML = 'FORTH';">FORTH</a>
</p>
<p>
Hello <b><span id="stuff">Stuff</span></b> there.
</p>
```

Updating the Browser Document

This is one reason why
you can only have one id
per document.

[BACK FORTH](#)

Hello **Stuff** there.

[BACK FORTH](#)

Hello **BACK** there.

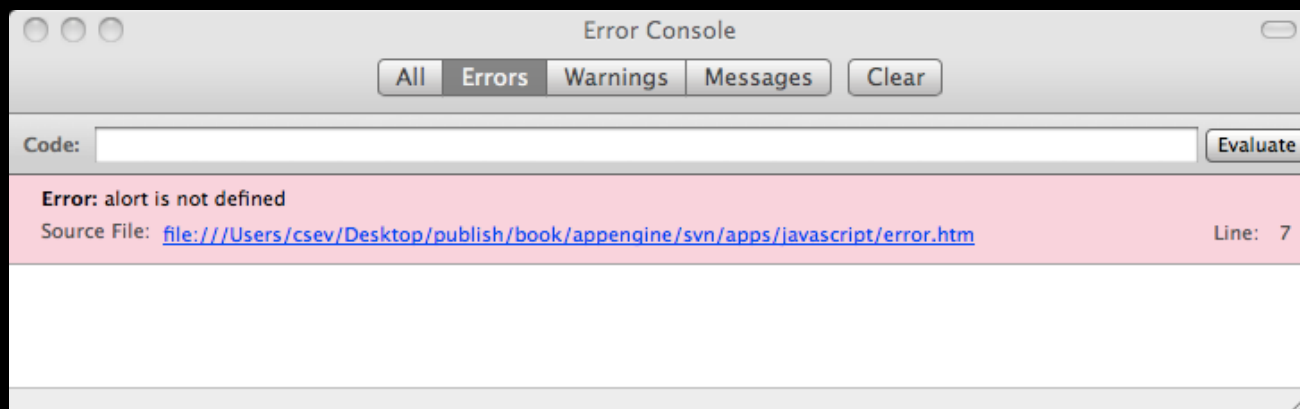
[BACK FORTH](#)

Hello **FORTH** there.

JavaScript Errors

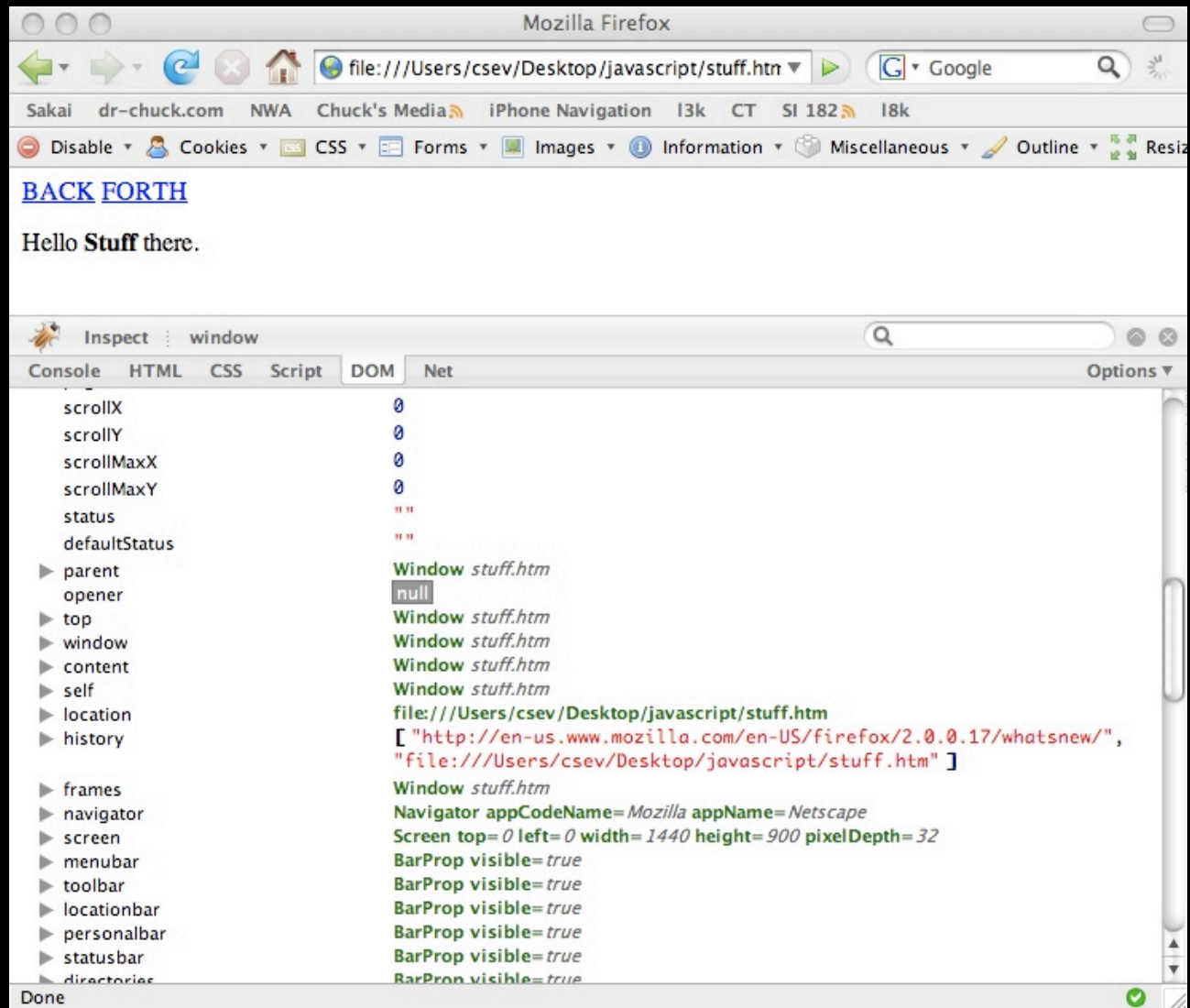
- Generally silently ignored by the browser
- Firefox: Tools -> Error Console

```
<html>
<body>
<h1>Here is my Document</h1>
<script type="text/javascript">
  alert("Hello from JavaScript");
</script>
<h1>Here is my second Header</h1>
</body>
</html>
```



FireBug - JavaScript / DOM

- If you are going to do **any** non-trivial JavaScript and/or DOM manipulation
 - Get FireBug for FireFox
 - addons.mozilla.org
- Many Javascript errors are **silent** - FireBug notices the errors



JavaScript Summary

- There is a lot of power in JavaScript - we keep “discovering” new uses and capabilities of JavaScript
- JavaScript is increasingly being treated as a very serious language
 - including using it in the server as well as the browser
- People now specialize as JavaScript developers

http://www.w3schools.com/js/js_examples.asp

JavaScript “Compilers”

- Some languages can be “compiled” into Javascript
 - Google Web Tool Kit - Java
 - Ruby - Red
 - Pyjamas - Python
- Google Chrome - Very Fast Javascript

Asynchronous JavaScript and XML (Ajax)

[http://en.wikipedia.org/wiki/Ajax_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming))

In The Good Old Days

- A user would take some action like a click on a link or button
- The Browser would make a TCP/IP connection to the web server
- The browser would send a POST or GET request
- The Server would send back a page to display to the user
- Repeat the Request-Response Cycle...

Browser

Redraw

Redraw

Redraw

Redraw

Whole
Page

Whole
Page

Whole
Page

Whole
Page

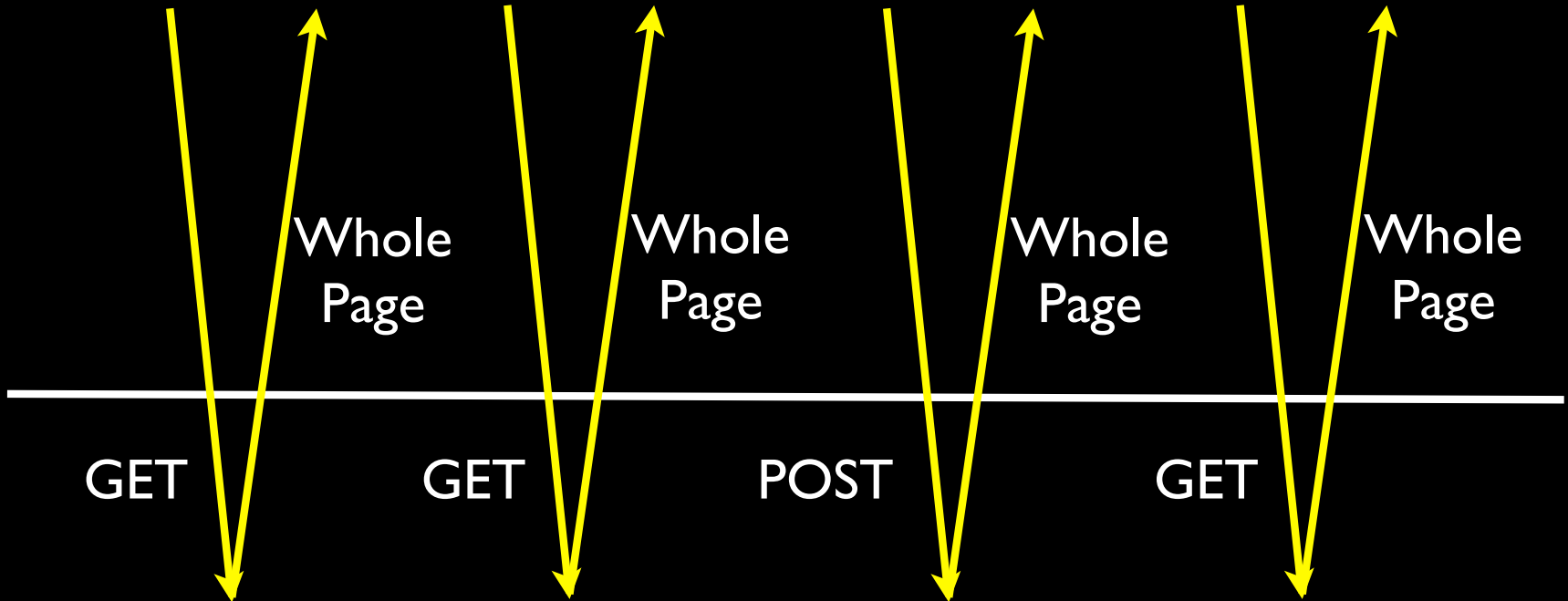
GET

GET

POST

GET

Server



XMLHttpRequest

- By 1999, **Microsoft** wanted to move some of the processing of web pages from the web server to the web browser
- The idea was instead of sending whole pages of HTML to the browser, send out the data to be displayed as XML and then produce presentation in JavaScript in the browser
- **Originally a Microsoft innovation** - other browsers soon adopted the idea and it became a defacto standard with a little variation between browsers :)
- It soon became clear that this could send *anything* - not just XML back and forth between a browser and client

<http://en.wikipedia.org/wiki/XMLHttpRequest>

Browser

Redraw

Update

Update

Update

Whole
Page

Fragment

Fragment

Fragment

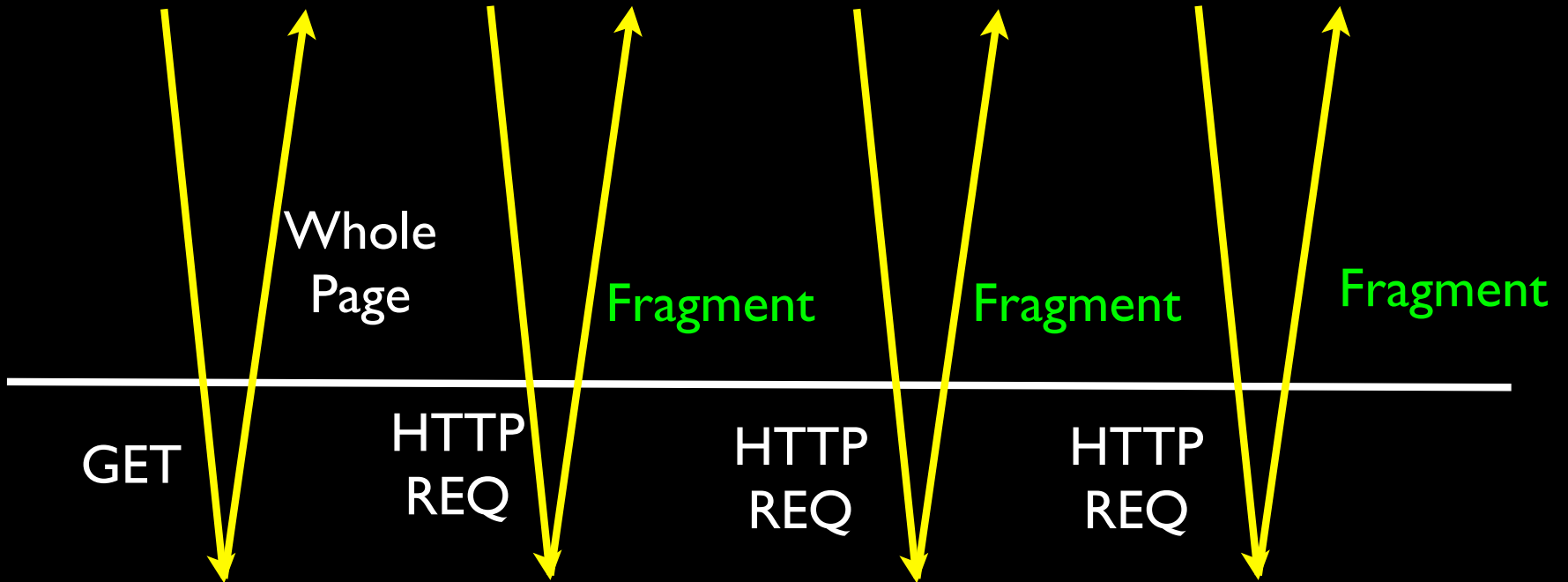
GET

HTTP
REQ

HTTP
REQ

HTTP
REQ

Server



Ajax Arms Race

- The race was on to build better and better web sites that began to replace things like frames and full-screen updates with XMLHttpRequest operations and very selective screen updates.
- With clever JavaScript programmers - the impossible became possible - drag and drop, automatic field completion - automatic data saving. It made the web operate much more like the desktop.
- Applications like GMail and Google Maps - feel very un-web.

Ajax versus Request/Response

- Standard Request/Response
 - Each click presents a whole new screen
- Ajax - Asynchronous JavaScript and XML
 - Each action sends data and receives results in the background.
 - The browser typically gets back a fragment of HTML or XML which is used to update a portion of the screen using the browser document model

Lots of Flexibility

- When you combine the ability to rewrite the Browser document model with the ability to interact with the web server from JavaScript - there is lots of potential fun
- Different browsers did things a *little* differently - this led to the rise of Ajax Libraries

Ajax Libraries

- Prototype - Basic portability and common functionality
 - <http://www.prototypejs.org/>
- Script.aculo.us - Animation and effects
 - <http://script.aculo.us/>
- jQuery - General purpose - <http://jquery.com/>
- Google Web Toolkit - <http://code.google.com/webtoolkit/>

Accessibility with Ajax

- It is not perfect
- Needs further study
- Lots of investment going on
- Fluid Project - Univ. Toronto
- <http://www.fluidproject.org/>



Google App Engine jQuery and Ajax

ae-l2-ajax

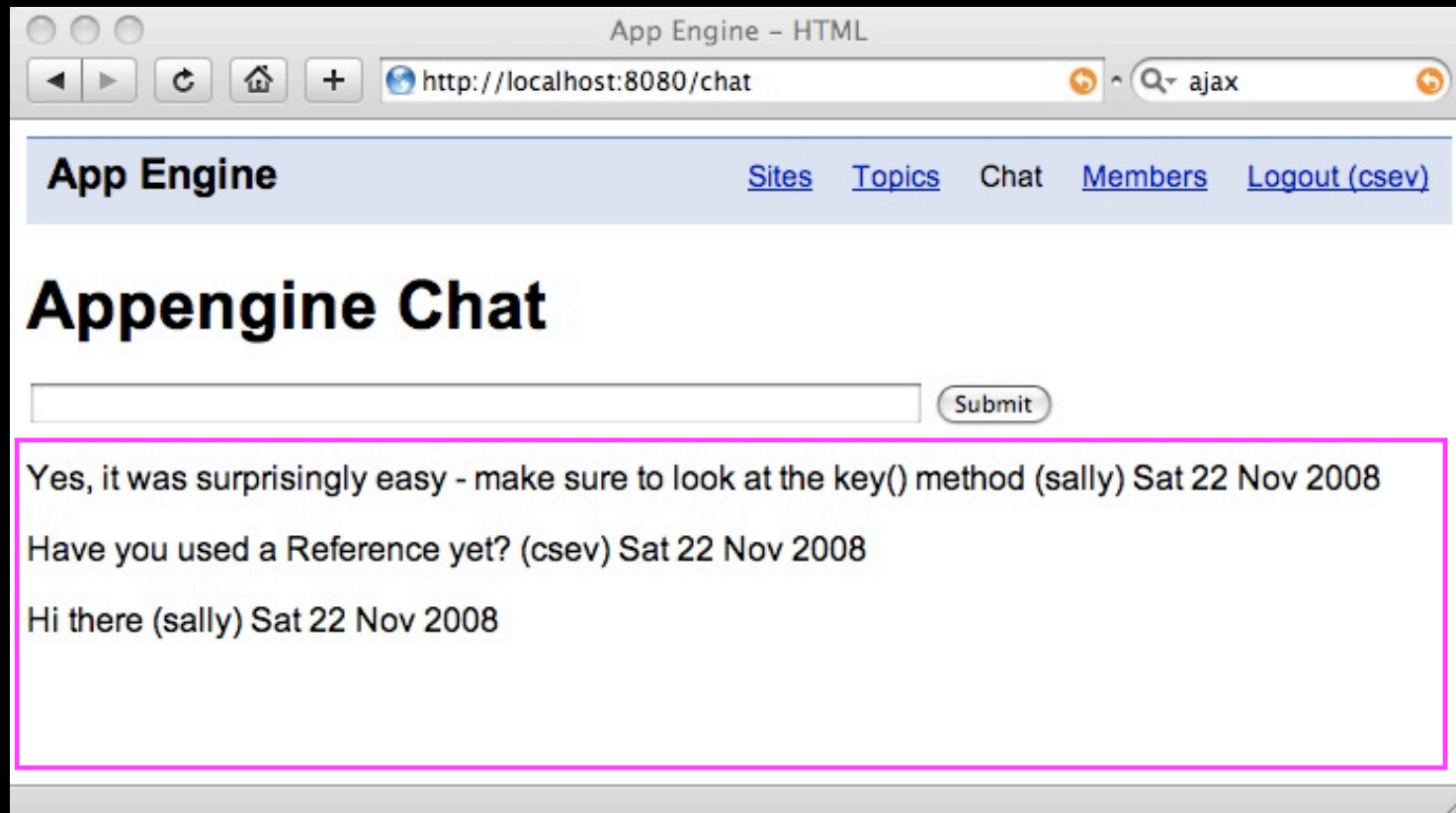
<http://ae-l2-ajax.appspot.com/>

Topics

- Installing jQuery and adding it to `_base.htm`
- Create new HTML fragment of messages at `/messages` - do not extend `_base.htm`
- Change `chat.htm` to use jQuery / Javascript and a timer

Ajax Enabled Chat

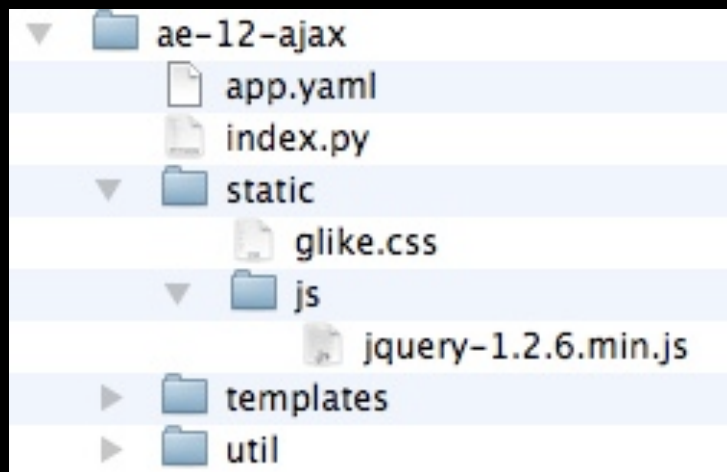
- We will update the list of messages in the background every four seconds
- This way we will see messages from other people “appear” on our screen - even if we are typing



We will put the chat messages in a div
and update the div using Ajax.

Intalling JQuery

- Go to jquery.com - download the latest version - source code
- Place it in your application under the **static** folder



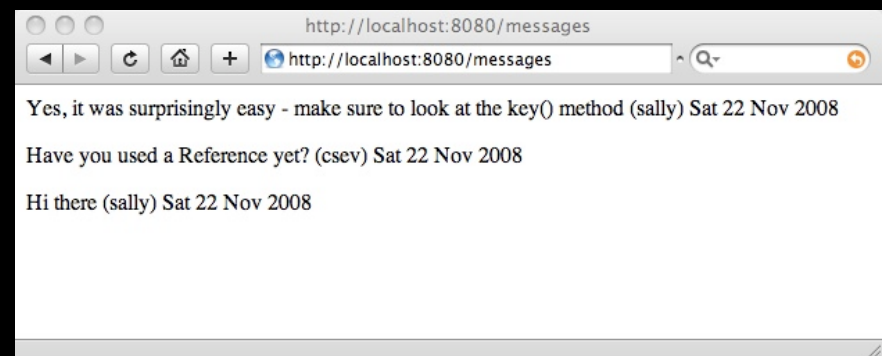
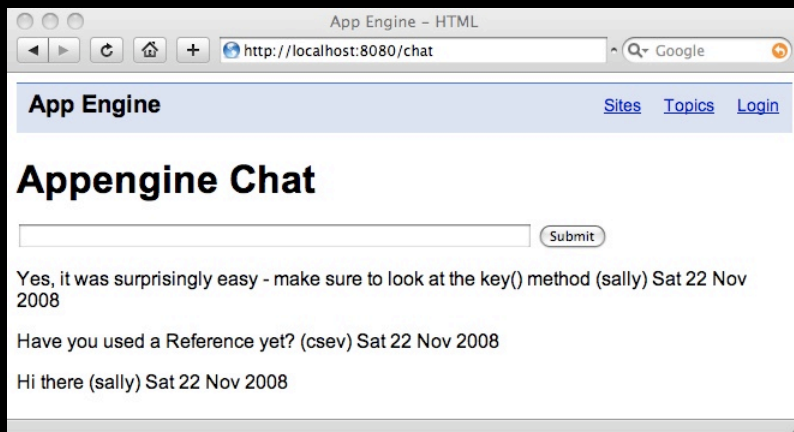
Installing jQuery

- To use the jQuery library in your HTML, you need to include it
- Typically this is done in `_base.htm` in the `<head>` area

```
<head>  
  <title>App Engine - HTML</title>  
  <link href="/static/glike.css" rel="stylesheet" type="text/css" />  
  <script type="text/javascript" src="/static/js/jquery-1.2.6.min.js"></script>  
</head>
```

HTML Fragments

- We need a page that just gives us message content
- No head material - not even a body tag



```
def main():  
    application = webapp.WSGIApplication([  
        ('/login', LoginHandler),  
        ('/logout', LogoutHandler),  
        ('/apply', ApplyHandler),  
        ('/members', MembersHandler),  
        ('/chat', ChatHandler),  
        ('/messages', MessagesHandler),  
        ('/*', MainHandler)],  
                                         debug=True)  
    wsgiref.handlers.CGIHandler().run(application)
```

New Routing Entry

```
class MessagesHandler(webapp.RequestHandler):  
  
    def get(self):  
        que = db.Query(ChatMessage).order("-created");  
        chat_list = que.fetch(limit=10)  
        doRender(self, 'messagelist.htm', {'chat_list': chat_list})
```

Retrieve the recent **the ChatMessage messages** and
put them in the **context** for out new **messages.htm**
template.

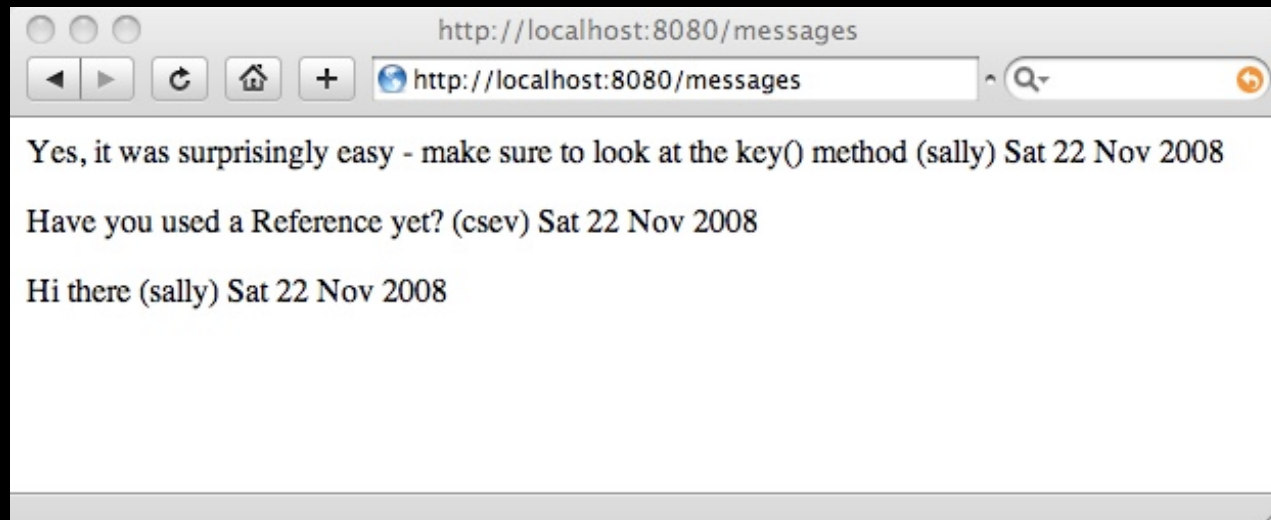
templates\messagelist.htm

```
{% for chat in chat_list %}  
  <p>{{ chat.text }} ({{chat.user.acct}})  
    {{chat.created|date:"D d M Y"}}</p>  
{% endfor %}
```

Do not extend `_base.htm` - we do not want the headers. Loop through the messages in the context and print out each message enclosed in a paragraph tag.

templates\messages.htm

```
{% for chat in chat_list %}  
  <p>{{ chat.text }} ({{chat.user.acct}})  
    {{chat.created|date:"D d M Y"}}</p>  
{% endfor %}
```



```

{% extends "_base.htm" %}
{% block bodycontent %}
    <h1>Appengine Chat</h1>
    <form method="post" action="/chat">
    <p>
    <input type="text" name="message"
        size="60"/>
    <input type="submit" name="Chat"/>
    </p>
    </form>
    {% ifnotequal error None %}
    <p>
    {{ error }}
    </p>
    {% endifnotequal %}

```

chatscreen.htm

```

    <div id="chatcontent">
        Loading...
    </div>
<script> /*  */
function updateMsg() {
$.ajax({
    url: "/messages",
    cache: false,
    success: function(frag){
        $("#chatcontent").html(frag);
    }
});
setTimeout('updateMsg()', 4000);
}
updateMsg();
/* ]]]&gt; */
&lt;/script&gt;
{% endblock %}
</pre>
</div>
```



```
<div id="chatcontent">
  Loading...
</div>
<script> /*  */
function updateMsg() {
$.ajax({
  url: "/messages",
  cache: false,
  success: function(frag){
    $("#chatcontent").html(frag);
  }
});
setTimeout('updateMsg()', 4000);
}
updateMsg();
/* ]]&gt; */ &lt;/script&gt;
{% endblock %}</pre></div><div data-bbox="484 220 936 346" data-label="Text"><p>The <b>chatcontent</b> div is where we will put the messages which we retrieve from the <b>/messages</b> url.</p></div><div data-bbox="490 437 932 602" data-label="Text"><p>The <b>$ajax()</b> call is from jQuery. It retrieves a <b>URL</b> and then looks up a <b>named div</b> and replaces its <b>html</b> content.</p></div><div data-bbox="504 684 929 767" data-label="Text"><p>Then we request that this be done every 4 seconds.</p></div>
```

```
INFO 2008-11-23 04:00:52,996 index.py] messages.htm
INFO 2008-11-23 04:00:53,016 dev_appserver.py] "GET /messages?_=1227412852966 HTTP/1.1" 200 -
INFO 2008-11-23 04:00:56,997 index.py] messages.htm
INFO 2008-11-23 04:00:57,016 dev_appserver.py] "GET /messages?_=1227412856968 HTTP/1.1" 200 -
INFO 2008-11-23 04:01:00,997 index.py] messages.htm
INFO 2008-11-23 04:01:01,013 dev_appserver.py] "GET /messages?_=1227412860970 HTTP/1.1" 200 -
INFO 2008-11-23 04:01:05,003 index.py] messages.htm
INFO 2008-11-23 04:01:05,021 dev_appserver.py] "GET /messages?_=1227412864972 HTTP/1.1" 200 -
INFO 2008-11-23 04:01:09,005 index.py] messages.htm
INFO 2008-11-23 04:01:09,028 dev_appserver.py] "GET /messages?_=1227412868976 HTTP/1.1" 200 -
```

You can watch the log as the Ajax requests
come in every 4 seconds.
The timestamp changes to make sure that the
pages are not cached.

```
class ChatHandler(webapp.RequestHandler):  
  
    def get(self):  
        que = db.Query(ChatMessage).order('-created');  
        chat_list = que.fetch(limit=10)  
        doRender(  
            self,  
            'chatscreen.htm',  
            { 'chat_list': chat_list })
```

The `chatscreen.htm` no longer needs the list of messages since they comes out in “/messages”.

```
class ChatHandler(webapp.RequestHandler):  
  
    def get(self):  
        que = db.Query(ChatMessage).order('-created');  
        chat_list = que.fetch(limit=10)  
        doRender(  
            self,  
            'chatscreen.htm')
```

The `chatscreen.htm` no longer needs the list of messages since they comes out in “/messages”.

Ajax Summary

- This is one of hundreds of Ajax techniques supported by JQuery
- It is a very common and useful pattern
- <http://www.jquery.com/> - much more detail